Statistically Significant Discriminative Patterns Searching

Hoang Son Pham, Gwendal Virlet, Dominique Lavenier, Alexandre Termier

Statistically Significant Discriminative Patterns Searching

Hoang Son Pham[1], Gwendal Virlet[2], Dominique Lavenier[2], and Alexandre Termier[2]

[1] ICTEAM, UCLouvain, Belgium
[2] Univ Rennes, Inria, CNRS, IRISA

Abstract. In this paper, we propose a novel algorithm, named SSDPS, to discover patterns in two-class datasets. The SSDPS algorithm owes its efficiency to an original enumeration strategy of the patterns, which allows to exploit some degrees of anti-monotonicity on the measures of discriminance and statistical significance. Experimental results demonstrate that the performance of the SSDPS algorithm is better than others. In addition, the number of generated patterns is much less than the number of the other algorithms. Experiment on real data also shows that SSDPS efficiently detects multiple SNPs combinations in genetic data.

Keywords: Discriminative patterns, Discriminative Measures, Statistical Significance, Anti-Monotonicity.

1 Introduction

Recently, the use of discriminative pattern mining (also known under other terms such as emerging pattern mining [1], contrast set mining [2]) has been investigated to tackle various applications such as bioinformatics [3], data classification [4]. In this paper, we propose a novel algorithm, named SSDPS, that discovers discriminative patterns in two-class datasets. This algorithm aims at searching patterns satisfying both discriminative scores and confidence intervals thresholds. These patterns are defined as **statistically significant discriminative patterns**. The SSDPS algorithm is based on an enumeration strategy in which discriminative measures and confidence intervals can be used as anti-monotonicity properties. These properties allow the search space to be pruned efficiently. All patterns are directly tested for discriminative scores and confidence intervals thresholds in the mining process. Only patterns satisfying both of thresholds are considered as the target output. According to our knowledge, there doesn't exist any algorithms that combine discriminative measures and statistical significance as anti-monotonicity to evaluate and prune the discriminative patterns.

The SSDPS algorithm has been used to conduct various experiments on both synthetic and real genomic data. As a result, the SSDPS algorithm effectively deploys the anti-monotonic properties to prune the search space. In comparison

with other well-known algorithms such as SFP-GROWTH [5] or CIMCP [6], the SSDPS obtains a better performance. In addition the proportion of generated patterns is much smaller than the amount of patterns output by these algorithms.

The rest of this paper is organized as follows: Section 2 precisely defines the concept of statistically significant discriminative pattern, and Section 3 presents the enumeration strategy used by the SSDPS algorithm. In Section 4, the design of the SSDPS algorithm is described. Section 5 is dedicated to experiments and results. Section 6 concludes the paper.

2 Problem Definition

The purpose of discriminative pattern mining algorithms is to find groups of items satisfying some thresholds. The formal presentation of this problem is given in the following:

Let I be a set of m items $I = \{i_1, ..., i_m\}$ and S_1, S_2 be two labels. A *transaction* over I is a pair $t_i = \{(x_i, y_i)\}$, where $x_i \subseteq I$, $y_i \in \{S_1, S_2\}$. Each transaction t_i is identified by an integer i, denoted *tid* (*transaction identifier*). A set of transactions $T = \{t_1, ..., t_n\}$ over I can be termed as a *transaction dataset* D over I. T can be partitioned along labels S_1 and S_2 into $D_1 = \{t_i \mid t_i = (x_i, S_1) \in T\}$ and $D_2 = \{t_i \mid t_i = (x_i, S_2) \in T\}$. The associated tids are denoted $D_1.tids$ and $D_2.tids$.

For example, Table 1 presents a dataset including 9 transactions (identified by 1..9) which are described by 10 items (denoted by $a..j$). The dataset is partitioned into two classes (class label 1 or 0).

Table 1: Two-class data example

Tids	**Items**										**Class**
1	a	b	c			f			i	j	1
2	a	b	c		e		g		i		1
3	a	b	c			f		h		j	1
4		b		d	e		g		i	j	1
5				d		f	g	h	i	j	1
6		b	c		e		g	h		j	0
7	a	b	c			f	g	h			0
8		b	c	d	e			h	i		0
9	a			d	e		g	h		j	0

A set $p \subseteq I$ is called an *itemset* (or pattern) and a set $q \subseteq \{1..n\}$ is called a *tidset*. For convenience we write a tidset $\{1, 2, 3\}$ as 123, and an itemset $\{a, b, c\}$ as *abc*. The number of transactions in D_i containing p is denoted by $|D_i(p)|$. The *relational support* of p in class D_i (denoted $sup(p, D_i)$), and *negative support* of p in D_i (denoted $\overline{\text{sup}}(p, D_i)$), are defined as follows:

$$sup(p, D_i) = \frac{|D_i(p)|}{|D_i|}; \quad \overline{\text{sup}}(p, D_i) = 1 - sup(p, D_i)$$

To evaluate the discriminative score of pattern p in a two-class dataset D, different measures are defined over the relational supports of p. The most popular discriminative measures are *support difference*, *grown rate support* and *odds ratio support* which are calculated by formulas 1, 2, 3 respectively.

$$SD(p, D) = sup(p, D_1) - sup(p, D_2) \tag{1}$$

$$GR(p, D) = \frac{sup(p, D_1)}{sup(p, D_2)} \tag{2}$$

$$ORS(p, D) = \frac{sup(p, D_1)/\overline{\text{sup}}(p, D_1)}{sup(p, D_2)/\overline{\text{sup}}(p, D_2)} \tag{3}$$

A pattern p is discriminative if its score is not less than a given threshold α. For example, let $\alpha = 2$ be the threshold of growth rate support. Pattern abc is discriminative since $GR(abc, D) = 2.4$.

Definition 1. ***(Discriminative pattern).*** *Let α be a discriminative threshold, $scr(p, D)$ be the discriminative score of pattern p in D. The pattern p is discriminative if $scr(p, D) \geq \alpha$.*

In addition to the discriminative score, to evaluate the statistical significance of a discriminative pattern we need to consider the confidence intervals (CI). Confidence intervals are the result of a statistical measure. They provide information about a range of values (lower confidence interval (LCI) to upper confidence interval (UCI)) in which the true value lies with a certain degree of probability. CI is able to assess the statistical significance of a result [7]. A confidence level of 95% is usually selected. It means that the CI covers the true value in 95 out of 100 studies.

Let $a = |D_1(p)|$, $b = |D_1| - |D_1(p)|$, $c = |D_2(p)|$, $d = |D_2| - |D_2(p)|$, the 95% LCI and UCI of GR are estimated as formulas 4 and 5 repectively.

$$LCI_{GR} = e^{\left(ln(GR) - 1.96\sqrt{\frac{1}{a} - \frac{1}{a+b} + \frac{1}{c} - \frac{1}{c+d}}\right)} \tag{4}$$

$$UCI_{GR} = e^{\left(ln(GR) + 1.96\sqrt{\frac{1}{a} - \frac{1}{a+b} + \frac{1}{c} - \frac{1}{c+d}}\right)} \tag{5}$$

Similarly, the 95% LCI and UCI of OR are estimated as formulas 6 and 7 repectively.

$$LCI_{ORS} = e^{\left(ln(ORS) - 1.96\sqrt{\frac{1}{a} + \frac{1}{b} + \frac{1}{c} + \frac{1}{d}}\right)} \tag{6}$$

$$UCI_{ORS} = e^{\left(ln(ORS) + 1.96\sqrt{\frac{1}{a} + \frac{1}{b} + \frac{1}{c} + \frac{1}{d}}\right)} \tag{7}$$

For example, consider the pattern abc in the previous example, the 95%CI of GR are $LCI_{GR} = 0.37, UCI_{GR} = 16.60$. Thus the GR score of abc is statistically significant because this score lies between LCI and UCI values.

Definition 2. ***(Statistically significant discriminative pattern).*** *Given a discriminance score $scr \in \{GR, ORS\}$, a discriminative threshold α and a lower confidence interval threshold β, the pattern p is statistically significant discriminative in D if $scr(p, D) \geq \alpha$ and $lci_{scr}(p, D) > \beta$.*

Problem statement: Given a two-class dataset D, a discriminance score scr and two thresholds α and β, the problem is to discover the complete set of patterns P that are statically significant discriminative for dataset D, discriminative measure scr, discriminative threshold α and lower confidence interval threshold β.

3 Enumeration Strategy

The main practical contribution of this paper is SSDPS, an efficient algorithm for mining statistically significant discriminative patterns. This algorithm will be presented in the next section (Section 4).

SSDPS owes its efficiency to an original enumeration strategy of the patterns, which allows to exploit some degree of anti-monotonicity on the measures of discriminance and statistical significance. In pattern mining enumeration strategies, *anti-monotonicity properties* is an important component. When enumerating frequent itemsets, one can notice that if an itemset p is unfrequent ($sup(p, D) < min\_sup$), then no super-itemsets $p' \supset p$ can be frequent (necessarily $sup(p', D) < sup(p, D) < min\_sup$). This allows to stop any further enumeration when an unfrequent itemset p is found, allowing a massive reduction in the search space [8]. As far as we know, no such anti-monotonicity could be defined on measures of discriminance or statistical significance.

The enumeration strategy proposed in SSDPS also builds an enumeration tree. However, it is based on the tidsets and not the itemsets. Each node of the enumeration tree is a tidset (with the empty tidset at the root). For example, consider the node represented by $\boxed{12:8}$ in Figure 1: this node corresponds to the tidset 128 in which $12 \subset D_1.tids$, and $8 \subset D_2.tids$.

Before presenting details of the enumeration strategy we first explain how to recover the itemsets from the tidsets. This is a well known problem: itemsets and tidsets are in facts dual notions, and they can be linked by two functions that form a *Galois connection* [9]. The main difference in our definition is that the main dataset can be divided into two parts ($D = D_1 \cup D_2$), and we want to be able to apply functions of the Galois connection either in the complete dataset D or in any of its parts D_1 or D_2.

Definition 3 (Galois connection). *For a dataset $D = D_1 \cup D_2$:*

- *For any tidset $q \subseteq \{1..n\}$ and any itemset $p \subseteq I$, we define:*

$$f(q, D) = \{i \in I \mid \forall k \in q \;\; i \in t_k\}; \quad g(p, D) = \{k \in \{1..n\} \mid p \subseteq t_k\}$$

- *For any tidset $q_1 \subseteq D_1.tids$ and any itemset $p \subseteq I$, we define:*

$$f_1(q_1, D_1) = \{i \in I \mid \forall k \in q_1 \;\; i \in t_k\}; \quad g_1(p, D_1) = \{k \in D_1 \mid p \subseteq t_k\}$$

- *For any tidset $q_2 \subseteq D_2.tids$ and any itemset $p \subseteq I$, we define:*

$$f_2(q_2, D_2) = \{i \in I \mid \forall k \in q_2 \;\; i \in t_k\}; \quad g_2(p, D_2) = \{k \in D_2 \mid p \subseteq t_k\}$$

Note that this definition marginally differs from the standard definition presented in [9]: here for convenience we operate on the set of tids $\{1..n\}$, whereas the standard definition operates on the set of transaction $\{t_1, ..., t_n\}$.

In Figure 1, under each tidset q, its associated itemset $f(q, D)$ is displayed. For example for node 12:8, the itemset $f(128, D) = bci$ is displayed. One can verify in Table 1 that bci is the only itemset common to the transactions t_1, t_2 and t_8. A *closure operator* can be defined over the use of the Galois connection.

Definition 4 (Closure operator). *For a dataset D and any tidset $q \subseteq \{1..n\}$, the closure operator is defined as: $c(q, D) = g \circ f(q, D)$. The output of $c(q, D)$ is the tidset of the closed itemset having the smallest tidset containing q.*

We can similarly define $c_1(q_1, D_1) = g_1 \circ f_1(q_1, D_1)$ for $q_1 \subseteq D_1.tids$ and $c_2(q_2, D_2) = g_2 \circ f_2(q_2, D_2)$ for $q_2 \subseteq D_2.tids$.

The basics of the enumeration have been given: the enumeration proceeds by augmenting tidsets (starting from the empty tidset), and for each tidset function f of the Galois connection gives the associated itemset. The specificity of our enumeration strategy is to be designed around statistically significant discriminative patterns. This appears first in our computation of closure: we divide the computation of closure in the two sub-datasets D_1 and D_2. This intermediary step allows some early pruning. Second, most measures of discriminance require the pattern to have a non-zero support in D_2 (GR and ORS). The same condition apply for measures of statistical significance: in both cases we need to defer measures of interest of patterns until it has some tids in D_2. Our enumeration strategy thus operates in two steps:

1. From the empty set, it enumerates closed tidsets containing only elements of D_1 (case group).
2. For each of those tidset containing only tids of D_1, augmentations using only tids of D_2 are generated and their closure is computed. Any subsequent augmentation of such nodes will only be allowed to be augmented by tids of D_2.

More formally, let $q \subseteq \{1..n\}$ be a tidset, with $q = q^+ \cup q^-$, where $q^+ \subseteq D_1.tids$ and $q^- \subseteq D_2.tids$. Then the possible augmentations of q are:

- (*Rule 1*) if $q^- = \emptyset$: q can either:
  - (*Rule 1a*) be augmented with $k \in D_1.tids$ such that $k < min(q^+)$
  - (*Rule 1b*) be augmented with $k \in D_2.tids$
- (*Rule 2*) if $q^- \neq \emptyset$: q can only be augmented with tid $k \in D_2.tids$ such that $k < min(q^-)$

This enumeration strategy allows to benefit from an anti-monotonicity property on the measures of statistical significance and discriminance.

Theorem 1 (Anti-monotonicity). *Let q_1 and q_2 be two tidsets such as: $q_1^+ = q_2^+$ and $q_1^- \subset q_2^-$ (we have $q_1^+ \neq \emptyset$ and $q_2^- \neq \emptyset$). Let $p_1 = f(q_1, D)$ and $p_2 = f(q_2, D)$. Then:*

1. $scr(p_1, D) > scr(p_2, D)$ *with scr a discriminance measure in* $\{SD, GR, ORS\}$.
2. $lci(p_1, D) > lci(p_2, D)$ *with lci a lower confidence interval in* $\{LCI_{ORS}, LCI_{GR}\}$.

Please refer to the full paper at *https://arxiv.org/abs/1906.01581* for the detailed demonstration of this part.

This theorem provides pruning by anti-monotonicity in our enumeration strategy: for a node having a tidset with tids both from $D_1.tids$ and $D_2.tids$, if the discriminance or statistical significance measures are below a given threshold, then necessarily its augmentations will also be under the threshold. Hence this part of the enumeration tree can be pruned. For example, node 2:8 has associated itemset $bcei$, and $ORS(bcei, D) = 3/4$. Suppose the ORS threshold $\alpha = 2$ this node can be pruned and its augmentations need not be computed. This allows to significantly reduce the search space.

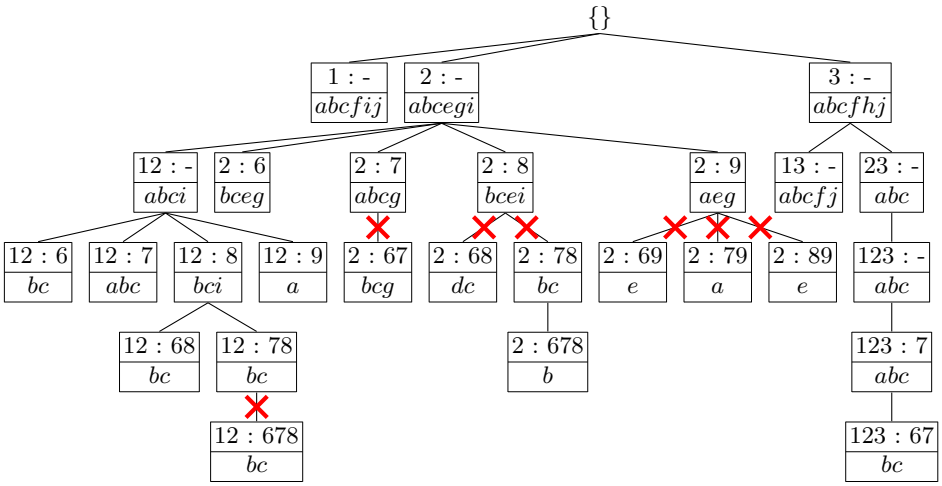

Fig. 1: Tidset-itemset search tree

4 SSDPS: Algorithm Design

This section presents the SSDPS algorithm which exploits the enumeration strategy presented in the Section 3.

As mentioned in the previous section, our algorithm is based on an enumeration of the tidsets. It discovers statistically significant discriminative closed patterns. The main procedure for enumerating tidsets is given in Algorithm 1. This procedure calls the recursive procedure `positiveExpand` (Algorithm 2) to find closed frequent itemsets in the positive class. Computing discriminative patterns relies on the recursive procedure `negativeExpand` (Algorithm 3).

Delving more into details, `positiveExpand` (Algorithm 2) is based on the principles of the LCM algorithm [10], the state of the art for mining closed frequent itemsets. `positiveExpand` takes as input the tidset t of a pattern that is closed in D_1 and a tid $e \in D_1.tids$ that can be used to augment t. This augmentation is performed on line 1, and the pattern p associated to the augmented tidset $t^+ = t \cup \{e\}$ is computed in line 2. If $p = \emptyset$, there are no items common

Algorithm 1: SSDPS algorithm

input : D, α, β
output: a set of statistically significant discriminative patterns

```
1  t = ∅
2  for each transaction id e in positive_class do
3      positiveExpand(t, e, D, α, β)
```

to all transactions of t^+ so the enumeration can stop (test of line 3). Else, we can continue the enumeration by applying *Rule 1* of enumeration presented in Section 3. Lines 4 to 9 apply the LCM principles of enumerating closed itemsets without redundancies (the interested reader in referred to [11] Section 3.2 for a recent description of these principles). At this step of the enumeration, the closure is computed in D_1 (line 4). The test of line 5 verifies if the closure actually extends the tidset, requiring a further verification in line 6, and the construction of the new extended tidset (line 7).

Lines 8 to 10 implement *Rule 1a* of enumeration, allowing to grow the positive part of the tidset. Lines 11 to 12 implement *Rule 1b* of enumeration, stopping the growth of the positive part and starting to grow the negative part of the tidset. The same logic is followed in lines 14 to 18, in the case where the tidset is not extended by the closure (test of line 5 is false).

Algorithm 2: positiveExpand Function

```
1  t⁺ ← t ∪ {e}
2  p ← f(t⁺, D)
3  if p is not empty then
4      t_ext⁺ ← c1(t⁺, D1)
5      if t_ext⁺ ≠ t⁺ then
6          if max(t_ext⁺) < e then
7              q ← t⁺ ∪ t_ext⁺
8              for each e⁺ in D1.tids \ q do
9                  if e⁺ < e then
10                     positiveExpand(q, e⁺, D, α, β)
11             for each e⁻ in D2.tids do
12                 negativeExpand(q, e⁻, D, α, β)
13     else
14         for each e⁺ in D1.tids do
15             if e⁺ < min(t⁺) then
16                 positiveExpand(t⁺, e⁺, D, α, β)
17         for each e⁻ in D2.tids do
18             negativeExpand(t⁺, e⁻, D, α, β)
```

The final expansion of the tidset is handled by `negativeExpand` (Algorithm 3), that can only perform augmentations with negative tidsets. It is very similar

to `positiveExpand`, with several key differences. The first obvious one is that the closure is this time computed in D_2 (line 4). The second one is that only *Rule 2* of enumeration can be applied (lines 13 and 20). The third and most important difference is that because we have tidsets with positive and negative tids, we can compute discriminance as well as statistical significance measures. Hence, Theorem 1 can be applied to benefit from pruning by anti-monotonicity. This is done in line 3.

Algorithm 3: negativeExpand Function

```
1  t⁻ ← t ∪ {e} ;    p ← f(t⁻, D)
2  if p is not empty then
3      if check_significance(p, D, α, β) is true then
4          t_ext⁻ ← c2(t⁻, D2)
5          if t_ext⁻ ≠ t⁻ then
6              if max(t_ext⁻) < e then
7                  q ← t⁻ ∪ t_ext⁻ ;    q_ext ← c(q, D);    p' ← f(q, D)
8                  if q_ext is empty then
9                      if check_significance(p', D, α, β) is true then
10                         output: p'
11                 for each e⁻ ∈ D2.tids \ q do
12                     if e⁻ < e then
13                         negativeExpand(q, e⁻, D, α, β)
14         else
15             t_ext ← c(t⁻, D)
16             if t_ext is empty then
17                 output: p
18             for each e⁻ ∈ D2.tids \ t⁻ do
19                 if e⁻ < e then
20                     negativeExpand(t⁻, e⁻, D, α, β)
```

5 Experimental Results

This section presents various experiments to evaluate the performance of the SSDPS algorithm. In addition, we apply the SSDPS to discover multiple SNPs combinations in a real genomic dataset. The details of the result was presented in the full paper which was published at *https://arxiv.org/abs/1906.01581*. All experiments have been conducted on a laptop with Core i7-4600U CPU @ 2.10GHz, 16GB memory and Linux operating system.

A synthetic two-class data was created to evaluate the pruning strategy as well as compare SSDPS with other algorithms. This dataset includes 100 transactions (50 transactions for each class). Each transaction contains 262 items which are randomly set by value 0 or 1. The density of data is set up to 33%.

5.1 Pruning Efficiency Evaluation

To evaluate the pruning efficiency of the SSDPS algorithm, we executed 2 setups on the synthetic dataset.

- Setup 1: use OR as discriminative measure; the discriminative threshold $\alpha = 2$.
- Setup 2: use OR as discriminative measure and LCI of OR as statistically significant testing; the discriminative threshold $\alpha = 2$, and LCI threshold $\beta = 2$.

As the result, the running time and the number of output patterns significantly reduce when applying LCI_{ORS}. In particular, with the setup 1, the SSDPS algorithm generates 179,334 patterns in 38.69 seconds while the setup 2 returns 18,273 patterns in 9.10 seconds. This result shows that a large amount of patterns is removed by using statistically significant testing.

5.2 Comparison with Existing Algorithms

We compare the performance of the SSDPS algorithm with two well-known algorithms: CIMCP [12] and SFP-Growth [5]. Note that these algorithms deploy discriminative measures which are different from the measures of SSDPS . In particular, CIMCP uses one of measures such as chi-square, information-gain and gini-index as a constraint to evaluate discriminative patterns while SFP-GROWTH applies $-log(p\_value)$. For this reason, the number of output patterns and the running times of these algorithms should be different. It is hence not fair to directly compare the performance of SSDPS with these algorithms. However, to have an initial comparison of the performance as well as the quantity of discovered patterns, we select these algorithms.

We ran three algorithms on the same synthetic data. The used parameters and results are given in Table 2.

Table 2: Used parameters and results of 3 algorithms

Algorithms	**Measure**	**Threshold**	**#Patterns**	**Time(seconds)**
SSDPS	OR, LCI_ORS	$\alpha = 2$, $\beta = 2$	49,807	73.69
CIMCP	Chi-square	2	5,403,688	143
SFP-GROWTH	-log(p_value)	3	*	> 172 (out of memory)

As the result, the SSDPS algorithm finds 49,807 patterns in 73.69 seconds; CIMCP discovers 5,403,688 patterns in 143 seconds. The SFP-GROWTH runs out of storage memory after 172 seconds. Hence the number of patterns isn't reported in this case.

In comparison with these algorithms the SSDPS gives a comparable performance, while the number of output patterns is much smaller. The reason is that the output patterns of SSDPS are tested for statistical significance by CI while other algorithms use only the discriminative measure. However, this amount of patterns is also larger for real biological analysis. Thus, searching for a further reduced number of significant patterns should be taken into account.

6 Conclusion and Perspectives

In this paper we propose a novel algorithm, called SSDPS, that efficiently discover statistically significant discriminative patterns from a two-class dataset. The algorithm directly uses discriminative measures and confidence intervals as anti-monotonic properties to efficiently prune the search space. Experimental results show that the performance of the SSDPS algorithm is better than other discriminative pattern mining algorithms. However, the number of patterns generated by SSDPS is still large for manual analysis. To reduce the amount of patterns our first perspective is to investigate a heuristic approach. Another perspective is to apply statistical techniques such as minimum description length or multiple hypothesis testing in order to further remove uninteresting patterns.

References

1. G. Dong and J. Li, "Efficient mining of emerging patterns: Discovering trends and differences," in *Fifth ACM SIGKDD*, ser. KDD '99. New York, NY, USA: ACM, 1999, pp. 43–52.
2. S. Bay and M. Pazzani, "Detecting group differences: Mining contrast sets," *Kluwer Academic Publishers*, vol. 5, no. 3, pp. 213–246–, 2001.
3. H. Cheng, X. Yan, J. Han, and P. S. Yu, "Direct discriminative pattern mining for effective classification," ser. ICDE '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 169–178.
4. M. Garca-Borroto, J. Martnez-Trinidad, and J. Carrasco-Ochoa, "A survey of emerging patterns for supervised classification," *Springer Netherlands*, vol. 42, no. 4, pp. 705–721, 2014.
5. L. Ma, T. L. Assimes, N. B. Asadi, C. Iribarren, T. Quertermous, and W. H. Wong, "An almost exhaustive search-based sequential permutation method for detecting epistasis in disease association studies," *Genetic Epidemiology*, vol. 34, no. 5, pp. 434–443, 2010.
6. T. Guns, S. Nijssen, and L. D. Raedt, "Itemset mining: A constraint programming perspective," *Elsevier*, 2011.
7. J. A. Morris and M. J. Gardner, "Statistics in medicine: Calculating confidence intervals for relative risks (odds ratios) and standardised ratios and rates," *British Medical Journal*, vol. 296, no. 6632, pp. 1313–1316, May 1988.
8. R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," *SIGMOD Rec.*, vol. 22, no. 2, pp. 207–216, Jun. 1993.
9. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering frequent closed itemsets for association rules," ser. ICDT '99. London, UK, UK: Springer-Verlag, 1999, pp. 398–416.
10. T. Uno, M. Kiyomi, and H. Arimura, "Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets," in *Workshop Frequent Item Set Mining Implementations*, 2004.
11. V. Leroy, M. Kirchgessner, A. Termier, and S. Amer-Yahia, "Toppi: An efficient algorithm for item-centric mining," *Inf. Syst.*, vol. 64, pp. 104–118, 2017.
12. T. Guns, S. Nijssen, and L. De Raedt, "Itemset mining: A constraint programming perspective," *Artificial Intelligence*, vol. 175, no. 12, pp. 1951–1983, 2011.